Docket No. P18184 UTILITY PATENT

UNITED STATES APPLICATION FOR LETTERS PATENT

for

DISTRIBUTED CONTROL PLANE ARCHITECTURE FOR NETWORK ELEMENTS

by

Hormuzd M. Khosravi, Sanjay Bakshi, Manasi Deval,

Rajeev D. Muralidhar and Suhail Ahmed

filed

November 13, 2003

DISTRIBUTED CONTROL PLANE ARCHITECTURE FOR NETWORK ELEMENTS

BACKGROUND

As the popularity of the Internet and other networks grows, the link bandwidth has grown from megabit rates to gigabit rates, and new protocols have been introduced to support new services being deployed on the Internet. The growth of new protocols and their related control traffic increases the overhead of a network. This may lead to three areas of difficulty with a protocol: scalability, highly available protocols, and robustness to control plane denial of service (DoS) attacks.

The addition of new devices and protocols increase the number of interfaces in a network element. The protocol generates more control traffic because it has to communicate with an increasing number of peers via many different interfaces. New services introduced in the network increase control traffic by either adding new types of control traffic or introducing new control protocols. This leads to increased complexity and traffic volume in the control plane, such as on the control card for the network element. This impedes network and network element scalability.

The growing use of real-time services such as network telephony, video and audio distribution, has resulting in a higher need for constant connectivity. The loss of a link for even a second can result in a loss of a considerable amount of information. An event such as a link failing will trigger huge amounts of processing related to the link event on the protocols executing on the control plane. This may saturate the control plane processor and render it nearly inoperational, causing the applications that need high availability to fail.

In another problem related to control plane saturation, DOS attack generally sends replicated or bogus packets to the control plane. The attack seeks to overwhelm the control plane by sending spoofed control packets for a particular protocol, such a Border Gateway

Protocol (BGP). Because all of the control packets need to be authenticated, the control card is overwhelmed attempting to authenticate the packets. This interferes with the control plane processing of other tasks, resulting in the denial of service.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention may be best understood by reading the disclosure with reference to the drawings, wherein:

Figure 1 is a block diagram of an embodiment of a network topology.

Figure 2 is a block diagram of an embodiment of a distributed network element.

Figure 3 is a block diagram of embodiments of two distributed control plane architecture control points.

Figure 4 is a flowchart of an embodiment to provide distributed control plane processing.

Figure 5 is a flowchart of an embodiment of a method for providing a distributed control plane architecture enabled network element.

Figure 6 is a flowchart of an embodiment of a method to establish initial connection for a controller control plane protocol module.

Figure 7 is a flowchart of an embodiment of a method to establish initial connection for a worker control plane protocol module.

DETAILED DESCRIPTION OF THE EMBODIMENTS

Figure 1 shows a topology of an embodiment of a network. Initially, the network may have been comprised of four devices or elements, such as 10, 12, 14 and 16, communicating between themselves. As has happened with the Internet and other large-scale networks, each of the network devices provides communication to more and more user devices. Eventually, the network devices may become communication points between networks, such as the network connected to network device 160. As networks grow, more and more devices need to communicate with each other. Further, the network devices 10, 12, 14 and 16, such as

switches or routers, need to keep each other updated with the addition of more members to the network.

Similarly, as technology has evolved, more services have been added to the network. These new services require new protocols or extensions to existing protocols. The term protocol as used here means any routing or signaling protocol. In order to service the new protocols or the extensions, network devices need to add new interfaces. A typically network device in a current Internet Service Provider (ISP) may have hundreds of physical or virtual interfaces. The ISP network may have 300-500 network devices, resulting in a huge number of interfaces. This large number of interfaces per network element coupled with an increase in the volume and variety of control traffic for the new devices and services limits the scalability of the network because the control planes of the network devices become overwhelmed.

Each network device, or element, generally has three operational planes: control, forwarding and management. The control plane typically executes signaling, routing and other control protocols in addition to configuring the forwarding plane. The forwarding plane usually performs the packet processing operations such as IP forwarding or classification, at or close to line-rate, the speed of the connecting line. The forwarding plane typically consists of specialized application specific integrated circuits (ASICs) or programmable network processors. The control plane typically executes on a general-purpose processor. The management plane provides administrative interface into the overall system and may consist of both software executing on a general-purpose processor as well as probes and counters in the hardware.

An example of a network device with control and forwarding planes is shown in Figure 2. The network device 10 has a control plane 20, and multiple forwarding planes 30, 32, 34 and 36. The control plane and the forwarding planes communicate with each other through a backplane 38. The backplane may be a physical backplane as is seen in a

computer, with the control plane, or card, and the forwarding planes, or lines cards, all connected to the backplane. Additionally, the control plane and forwarding planes may not all physically reside on a physical backplane, but are connected through a backplane fabric, such as an Internet Protocol (IP) network, Asynchronous Transfer Mode (ATM), Ethernet, or some proprietary switching fabric. The backplane is used to transfer both control and data traffic between the planes.

The control plane processing is done in a centralized manner by the control card. The line cards, or forwarding planes, handle the media-specific processing and line-rate packet processing. The separation of processing tasks between the control and forwarding planes has advantages in that each can scale independently, and different components from different vendors can be used to build each plane because of the use of a standardized inter-plane interface, such as the Network Processing Forum application program interface.

This distributed processing architecture shown in Figure 2 allows the use of a distributed control plane architecture that can be used to distribute functionality of complex control plane protocols between different processing elements in network devices. Prior to embodiments of this invention, this type of distributed processing could not be accomplished because of the difficult of maintaining connectivity and tracking of the various protocols and processing tasks between the various processing elements.

Embodiments of the distributed control plane architecture (DCPA) have two modules that allow this distributed processing: the DCPA Infrastructure Module (DIM), and the DCPA Communications Library (DCL). The control plane and/or the forwarding planes that execute a DIM along with one or multiple routing or signaling protocols functions is referred to here as a control point. The DIM also contains the logic required by a control point to discover other control points present in a system, establish connectivity with them and exchange information about their capabilities. The DIM may also be configured with a policy or policies that determine the control points with which it will communicate.

The DIM maintains the current view of the network device, keeping track of the capabilities and resources of other control points in the network element and the Control Plane Protocol Modules (CPPMs) running on each of them. A CPPM is a protocol implementation that executes a singe function or the complete functionality of a protocol in accordance with a standard, such as an industry standard or an Internet Engineering Task Force (IETF) request for comments (RFC). Multiple CPPMs work in conjunction to provide a complete implementation of the protocol. The functionality may be split into a core functionality, more than likely executed by the control plane, referred to as a Controller CPPM, and a portion separated from the core functionality, referred to as a Worker CPPM.

The DCL is linked with the CPPMs and is responsible for providing communication between them. Abstraction of the CPPMs for communication purposes is provided to them by the transport abstraction. The peer CPPM information is provided by the DIM. Generally, the DIM and the CPPM will run in different processes and communicate via an interprocess communication.

For example, assume a protocol such as Open Shortest Path First (OSPF) was distributed with the OSPF controller CPPM running on the control plane and two OSPF worker CPPMs on the forwarding plane. The OSPF controller would consider the 2 workers as peer CPPMs. The common interface between the peer CPPMs is known as the CPPM Peer Interface.

Figure 3 shows an embodiment of two control points in a distributed control point architecture. In this particular embodiment, control point 1 is a control plane and control point 2 is a forwarding plane. These are merely designated for this example and no limitation is intended by this configuration. Control point 1, 40, has the DIM 42 and the DCL 44. The DCL is linked to the controller CPPM 46. While only one controller CPPM is shown, there will be a controller CPPM for each protocol being executed by the control point.

Turning to the DIM, it can be seen that the DIM is made up of a namespace 423, a binding and discovery module 422, the CPPM registration (CR) module 425, a packet redirection module (PR) 426, APIs 424 and a transport module 421. The namespace 423 maintains a logical view of the network element and its components. Various modules register with the namespace to receive notification of particular events when there are changes to the namespace. For example, a OSPF CPPM may register with the namespace to be notified if any other control points introduce another OSPF CPPM. When the new CPPM registers with the namespace, the existing CPPM is notified.

The binding and discovery module 422 discovers other control points and exchanges configuration information with them. The information may include properties of the control points and the capabilities of any CPPMs running on those control points. When the information is exchanged, the binding and discovery module then updates the information, or binding, in the namespace.

The CPPM Registration module (CR) 425 allows a CPPM to register its capabilities and get information about its peers. A CPPM uses the packet redirection module 426 to specify the protocol packets of interest to itself. For example, a worker CPPM may be executing part of the Open Shortest Path First (OSPF) protocol. If an update to the OSPF configuration comes in, this packet would be re-directed to the CPPM registered for it. If no such redirection were specified, the controller CPPM would receive all protocol packets by default.

One example of the interface that receives and routes packets may be found in co-pending US Patent Application No. 10/XXX,XXX, filed November 14, 2003, "Implementation Of Control Plane Protocols And Networking Stacks In A Distributed Network Device." Other methods of providing this interface could also be used, but this provides an example of one.

The APIs 424 provide a standardized interface to control protocol stacks. A protocol stack is a stack of network layers that work together. An example of a control protocol would be OSPF (Open Shortest Path First). Typically, these APIs would be standardized to match those defined by the Network Processing Forum (NPF).

The transport module 421 is responsible for communication between the DIMs. It hides the specifics of the physical interconnection and associated transport protocols used to exchange information between them. It provides a connection-oriented interface to set up connections and transfer data between any two DIMs in the system and takes care of managing the underlying transport protocols and interconnects. This transport module should be a 'plug-in,' such that it can be changed and updated without affecting any of the other modules. New plug-ins can be easily added to support additional interconnects and associated transport protocols.

The DIMs between the control plane control point 1 and the forwarding plane control point 2 are substantially the same. The DIMs exchange information about the CPPMs running on them. The DCLs 44 and 54 communicate with the DIMs 42 and 52, respectively, to discover peer CPPMs and provide abstractions of the peer CPPMs to the local CPPM. In control point 2 50, the CPPM 56 is local to the DCL 54 and the controller CPPM 46 is local to the DCL 44. The interface between the control and worker CPPMs is through the peer CCPM API 443.

The DCL provides an abstraction of the peer CPPMs in the form of the resources they own or control. The abstraction refers to the ability of the DCL to make the distribution transparent to the CPPMs. Using the OSPF controller and worker CPPMs above, the DCL uses the resource information to access connection information. This information is in turn used to send messages to the appropriate OSPF worker CPPMs. The OSPF controller 'believes' that it is communicating directly with the OSPF worker, making the distribution transparent.

The DCP messaging layer 442 controls the actual communication between the CCPMs across the peer CCPM interface. It provides functions to serialize and de-serialize data and transparently exchange it between the associated CPPMs across the network. The messaging layer uses information from the CPPM Registration module in the DIM to set up and manage communication channels between the associated CCPMs. The messaging layer detects CPPM and connection failures and is informed of changes in the configuration of peer CPPMs via events from the DIM. The DCL can implement load balancing, redundancy and dynamic failover, if those policies are configured.

The messaging layer relies upon the transport abstraction layer 441 to set up, manage communications and provide interconnect/transport independent communications. The abstraction again 'hides' the details of the interconnect and transport protocols. This allows for easy integration with existing protocol stacks.

Using the DIM and DCL components to manage control plane processing in devices, it is now possible for a device to handle increased control traffic without overwhelming the control plane. Portions of the control plane processing for a particular protocol can be off-loaded from the control plane to the forwarding planes, independent of protocols and interconnections used, as well as ease of adding new protocols and scaling to multiple processor resources.

This architecture allows the system designers to consider distribution control processing for protocols. An embodiment of this process is shown in Figure 4. During evaluation of the protocol for distribution at 60, there are typically 2 issues to be considered: what to distribute; and where to distribute.

Control protocols may be classified in many different ways. For the discussion here, they will be split into three categories: link-specific functions involving packet forwarding and maintenance of connectivity with protocol peers; protocol-specific processing such as maintaining the protocol state machine; and functions that update the forwarding plane with

control information. Link-specific functions generally involve parts of a protocol responsible for communication with peers to maintain adjacency and status of the connectivity, and processing of packets on a per interface basis. These functions are ideal candidates for distribution to on-line cards. This reduces the computation load on the control processor(s) and bandwidth load on the interconnect fabric, as well as accelerating peer discovery and failure detection.

Protocol-specific functions are typically related to computation, based upon updates received from the protocol peers to generate new peer updates and new control information, such as new IP routes for installation in the forwarding plane. The protocol state machine maintenance functions are included in this category. These functions could be considered for distribution on a case-by-case basis, with the focus being on reducing the computational load on the control processor when network failures cause additional message traffic.

Forwarding plane update or centralized functions consist of updates to refresh the control information in the forwarding plane. An example may include routing table updates as information comes in from adjacent devices about link failures or additions to the network. These functions can rarely be distributed.

For the functions that can be distributed, the DCPA manages the distribution and communication between the control processors and the forwarding plane processors. Returning to Figure 4, after a protocol has been selected for distribution, the protocol must be portioned out amongst the processing entities. In the DCPA, this would be accomplished by defining the control and worker Control Plane Processing Modules, mentioned earlier, at 62. The interfaces are then provided at 64 between the control CPPM and the worker CPPM to allow them to communicate about the protocol processing they are performing.

Once the interfaces are provided, the communications library entries, such as the code that allows the interfaces to operate, is defined at 66. The code or other implementation of the DIM, the DCL and the controller CPPM is then integrated onto the control card at 68 and

the forwarding plane at 69. Generally, the embodiments of this invention will take the form of machine-readable code that, when executed, cause the machine to distribute the processing tasks as discussed above. The machine in this context is the control plane processor or processors and the forwarding plane processors.

Once the control plane and forwarding planes are established, the network device can initiate communication with other network devices. An embodiment of this establishment is shown in Figure 5. Communications are initiated at 70. The controller components register with the namespace at 72 for the appropriate events, and the DIM discovers other control points at 74. The namespace is updated at 76 as each control point is discovered and the communications library is similarly initiated at 78. Finally, the CPPMs register with the CPPM at 80.

Within that process, the CPPMs initialize and set themselves up, such as in the embodiments of Figure 6 and 7. In Figure 6, a process for initial connection establishment for a controller CPPM is shown. When the control card initialized, such as on boot up, at 90, the control card registers with the DIM at 92. At 94, the controller CPPM waits until worker CPPM register. When a worker CPPM registers, the DCL sets up the connection between the worker and controller CPPM at 96.

Figure 7 shows the similar process for the worker CPPM. The worker CPPM is initialized at 91. It registers at 93 with the DIM. If the controller CPPM has not registered yet, the worker CPPM waits at 95. Once the controller CPPM has registered, the DCL sets up the control connections for the worker CPPM and the controller CPPM at 97.

In this manner, a generic software architecture is provided that can be used to distribute the functionality of complex control plane protocols between different processing elements in network elements. This architecture provides scalability, high availability and robustness without significantly increasing the cost of the network element. The architecture distributes and scales the control plane protocol functionality across different processing

elements/resources in the system or network element without any major re-design of the control protocol. It is independent of the control plane protocol and backplane interconnect technology, supports multiple processor hierarchies in the system and provides well-defined interfaces that allow new protocol implementations to be easily added into the system.

Thus, although there has been described to this point a particular embodiment for a method and apparatus for establishing a distributed control plane architecture, it is not intended that such specific references be considered as limitations upon the scope of this invention except in-so-far as set forth in the following claims.